Accelerated Q-Learning with Markov Chain Monte Carlo (MCMC) Metropolis-Hastings Sampling

A. Nair*

*Santa Susana High School, Simi Valley, USA (advaithnair476@gmail.com)

SUMMARY
Reinforcement learning (RL) is a machine learning paradigm where a symbolic agent interacts with a presented environment. It navigates the intrinsic state-action pairs to maximize cumulative rewards that are interspersed throughout it, eventually learning an optimal policy that maximizes the total collected reward. Q-learning is a foundational RL algorithm that quantifies this process by assigning numerical values to state-action pairs. However, it often struggles in large, continuous action spaces due to suboptimal exploration strategies (ε-greedy, Boltzmann exploration). To address these limitations, introducing Markov Chain Monte Carlo (MCMC) methods, such as the Metropolis-Hastings algorithm can improve reward collection and efficiency. MCMC-integrated Q-learning yields a nearly six-fold increase in collected rewards. These findings underscore the potential of MCMC-driven advancements in RL, prompting further research into complex environment navigation algorithms.

KEYWORDS
Artificial intelligence, Markov Chain Monte Carlo, Metropolis-Hastings, reinforcement learning

INTRODUCTION
RL is one of three primary machine learning paradigms. Unlike its counterparts supervised and unsupervised learning—which form conclusions based on labeled and unlabeled data, respectively—reinforcement learning is more abstract, showing a symbolic agent's traversal through a simulated virtual environment, through which the agent receives various rewards, or R (Sarker, 2021). If a model of the environment is provided to the agent, it is referred to as model-based; otherwise, it is model-free. An agent's strategy in pursuing rewards is known as its policy. Algorithms can be on-policy, where the agent learns and improves its current policy, or off-policy, where learning involves a separate behavior and decision policy.

Environments are broken into states (s), representing their current configuration, and timesteps, discrete intervals where changes occur or actions are taken. Actions (a) can be taken by the agent to traverse states, and the result is probabilistically dependent on whether the environment is stochastic or deterministic. These assumptions are formalized in a mathematical framework known as the Markov Decision Process (MDP). Aside from states, actions, and rewards, there also exists a discount factor $\gamma \in [0, 1]$ which devalues future rewards in favor of current and a learning rate $\alpha \in [0, 1]$ that determines how much new information overrides old information (Alagoz et al., 2010).

Q-learning is a model-free, off-policy RL algorithm that defines its optimal policy by storing the relative value of state-action pairs in a Q-table. These Q-values are progressively tweaked as the agent explores the environment by the Bellman optimality equation, seen below. As more training episodes are completed, these Q-values converge (Watkins & Dayan, 1992).

$$Q(s, a) = Q(s, a) + \alpha[R + \gamma Q(s', a') - Q(s, a)]$$

The convergence of traditional Q-learning algorithms is limited by large and continuous action spaces, as well as the challenges with uniform random exploration using policies like ε-greedy and Boltzmann exploration. Methods like these explore action spaces inefficiently due to the use of static, linear hyperparameters (Wei, 2024). In the case of ε-greedy, the choice between exploration and exploitation is based on the value of ε; the agent explores with probability ε and exploits with a likelihood of 1-ε (Dann et al., 2022).

This can be mitigated by using Markov Chain Monte Carlo (MCMC) methods; MCMC methods are a class of algorithms used to sample from probability distributions when direct sampling is difficult. A Markov chain is a stochastic process where the next state is dependent on only the previous state, and Monte Carlo methods involve using random sampling to estimate numerical results. The two can be combined with MCMC, where a Markov chain is constructed that has the desired distribution which

is created by using samples of states (Robert et al, 2018). In this paper, the efficacy of the MCMC Metropolis-Hastings theorem—a standard MCMC sampling method where policy changes are randomly rationalized for faster convergence— in conjunction with Q-learning is explored. Results will be compared to a traditional Q-learning algorithm's results.

METHODS

Environment Selection
The aim of this study was to compare the efficiency of traditional Q-learning and accelerated Q-learning for collecting rewards. Both algorithms were coded in Python, and extensively utilized Farama Foundation's Gymnasium library, which provides a graphical interface with many different RL environments that can be run and simulated.

Both the standard Q-learning and accelerated Q-learning models were run on the same environment: *Taxi-v3*. As part of the "Toy Text" classification of environments provided by the Farama Foundation, *Taxi-v3* has the largest overall state-action space; a small state-action space such as that of *FrozenLake-v1*, would likely see negligible differences between the two.

Parameter and Hyperparameter Selection
The parameters and hyperparameters selected for the two models were consistent. The MCMC Metropolis-Hastings algorithm utilizes a special hyperparameter known as the temperature (τ), which guides the locations of sampling and the acceptance of certain actions over others. This was not included in the standard Q-learning model.

Table 1.1: Parameter and Hyperparameter Comparison

Parameter/Hyperparameter	Standard Q-learning	MCMC-accelerated Q-learning
Learning rate (α)	0.25	0.25
Epsilon for epsilon-greedy (ε)	0.1	0.1
Discount factor (γ)	0.95	0.95
Temperature (τ)	N/A	1.0
# of training episodes	50000	50000

RESULTS AND DISCUSSION
Five 50000-episode reward totals were recorded for both the standard Q-learning algorithm and the accelerated Q-learning algorithms. The mean was taken to obtain a conclusive metric of the models' performance. The MCMC Metropolis-Hastings algorithm was significantly more efficient than the standard Q-learning model, as seen in Table 1.2.

Table 1.2: Reward Comparison between the Standard Model and Accelerated Model

Model	Mean five-trial reward over 50000 episodes
Standard Q-learning	60263.8
MCMC-accelerated Q-learning	330590.2

The primary reason that the MCMC-accelerated model performed more efficiently than the standard model was the remarkable consistency with which it sampled high-reward areas of the environment. Unlike the standard model, which fluctuated between high and low-reward areas, the accelerated

model was able to identify high-reward areas and stay there, due to the Metropolis-Hastings sampling distribution. .

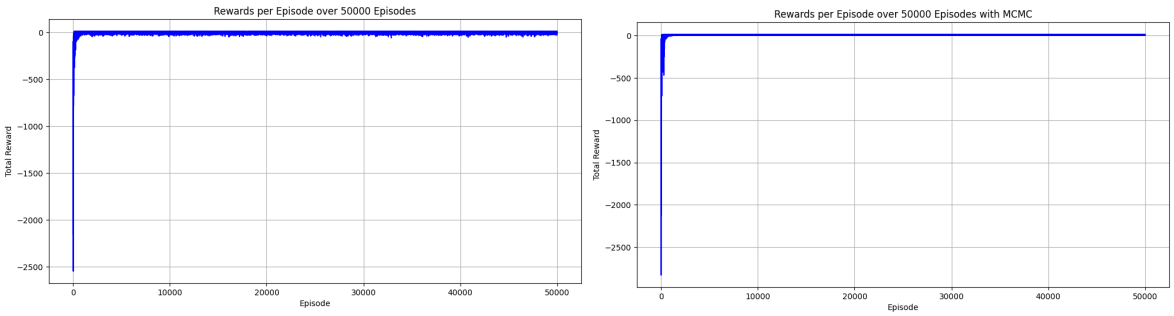

Figure 1.2: Episodic Convergences of Standard Model (left) and Accelerated Model (right)

The consistency of the accelerated model can be attributed to the implementation of Metropolis-Hastings acceptance probability and the temperature τ parameter. The probability accounts for all Q-values before and after a proposed action; if the ratio of proposed to current is greater than one, it is always accepted, analogous to a greedy algorithm. However, if the ratio is less than one, the action isn't immediately denied, but rather accepted with a τ-dependent probability, where higher τ values means the model is more risky and vice versa.

When this Metropolis-Hastings acceptance probability is run together with an ε-greedy policy, it leads to greater stability and higher rewards.

CONCLUSIONS

The purpose of this paper was to investigate the efficiency of an accelerated Q-learning algorithm that leverages MCMC Metropolis-Hastings sampling. When tested with consistent hyperparameters over a 50000-episode training cycle, the MCMC-accelerated algorithm saw nearly a six-fold increase in total rewards obtained. This can largely be attributed to the temperature parameter τ which gave the model extra stability in the exploration-exploitation process.

These promising results with MCMC algorithm integration into RL algorithms can spur further research; Gibbs sampling and Hamiltonian dynamic Monte Carlo methods, two other MCMC algorithms,would likely see promising results within RL. Furthermore, on-policy algorithms like Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO) could be enhanced with MCMC sampling, another area for future work.

REFERENCES

1. Sarker, I.H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(160). https://doi.org/10.1007/s42979-021-00592-x
2. Alagoz, O., Hsu, H., Schaefer, A. J., Roberts, M. S. (2010). Markov Decision Processes: A Tool for Sequential Decision Making under Uncertainty. *Medical Decision Making*, 30(4), 474-483. https://doi.org/10.1177/0272989X09353194
3. Watkins, C.J.C.H. & Dayan, P. (1992). Q-learning, *Machine Learning*, 8, 279-292. https://doi.org/10.1007/BF00992698
4. Wei, D. (2024, May 13). *Demystifying Q-Learning*. Medium. https://medium.com/@weidagang/demystifying-q-learning-32c867a105de
5. Robert, C.P., Elvira, V., Tawn, N. & Wu, C. (2018). Accelerating MCMC algorithms. *WIREs Computational Statistics*, 10(5), https://doi.org/10.1002/wics.1435
6. Dann, C., Mansour, Y., Mohri, M., Sekhari, A. & Sridharan, K. (2022). Guarantees for Epsilon-Greedy Reinforcement Learning with Function Approximation. https://doi.org/10.48550/arXiv.2206.09421